Extension of EllipSys to compressible flows - Implementation and verifications

Bertagnolio, Franck; Sørensen, Niels N.

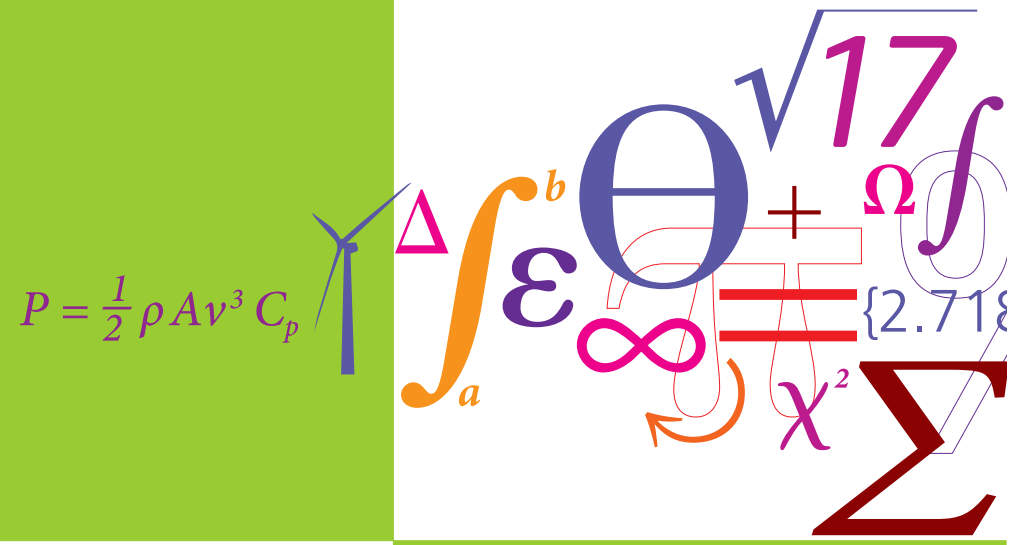
Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Extension of EllipSys to compressible flows

Implementation and verifications

Report E-0170



Franck Bertagnolio and Niels N. Sørensen
DTU Wind Energy
June 2018

Extension of EllipSys to compressible flows

Implementation and verifications
2018

Af
Franck Bertagnolio and Niels N. Sørensen

Copyright: -
Front picture: -
Published by: DTU Wind Energy, Frederiksborgvej 399, DK-4000 Roskilde
Requests at: info@vindenergi.dtu.dk

ISSN: 0000-0000 (electronic version)
ISBN: 978-87-93549-38-8 (electronic version)

ISSN: 0000-0000 (printed version)
ISBN: 000-00-0000-000-0 (printed version)

Author(s): Franck Bertagnolio and Niels N. Sørensen

Title: Extension of EllipSys to compressible flows - Implementation and verifications

Department: DTU Wind Energy

Summary (max. 2000 characters):
This report gives a brief overview of the main implementation details necessary to account for compressibility in the EllipSys code. The additional input parameters required to activate and configure the compressible formulation are presented. A series of test-cases are defined and comparisons with measurement data and solutions from existing validated compressible codes are conducted. It is shown that the code behaves as expected, although it typically requires longer computational times than for the incompressible case. Furthermore, the quantitative results appear to validate this compressible version of the code.

Report number: DTU Wind Energy Report E-0170

Publication date: June 2018

Contract no.:
Project no.:
Sponsorship: AVATAR (European Union grant agreement No. FP7-ENERGY-2013-1/no.608396, and 'Rain Erosion Tester for Accelerated Test of Wind Turbines' (Energistyrelsen, EUDP, Jr. nr. 64015-0045)

Front page:

Reference: (Electronic)
ISSN: 0000-0000
ISBN: 978-87-93549-38-8

Reference: (Print)
ISSN: 0000-0000
ISBN: 000-00-00000-00-0

Pages: 19
Tables: 0
Figures: 0
References: 0

Technical University
of Denmark
DTU Wind Energy
Frederiksborgvej 399
4000 Roskilde
Denmark

Preface

As wind turbines are steadily getting bigger, their blades' radii become very large, of the order of 100 m for the largest ones to these days. Combined with the rotor rotational velocity required to create lift on the blades for generating torque and thereby electricity, modern wind turbines operate at such blade tip velocities that compressibility effects start to play a role on the blade aerodynamics. EllipSys being an incompressible flow solver by design and being partly dedicated to compute wind turbine rotor flows, it has become important to extend its capabilities to compressible flows.

Content

Summary

This report gives a brief overview of the main implementation details necessary to account for compressibility in the EllipSys code. The additional input parameters required to activate and configure the compressible formulation are presented. A series of test-cases are defined and comparisons with measurement data and solutions from existing validated compressible codes are conducted. It is shown that the code behaves as expected, although it typically requires longer computational times than for the incompressible case. Furthermore, the quantitative results appear to validate this compressible version of the code.

1 Introduction

The aim of this report is to:

1. give an overview of the implementation of compressibility in the 2D versions of EllipSys, and
2. verify the sanity of the code using a number of test-cases based on experimental data and by comparison with existing and validated compressible codes.

The next section is dedicated to the code implementation of compressibility and modifications to the code are briefly reviewed. The third section is orientated toward the user-interface and the new keywords available to the user for activating compressibility features are presented. The fourth section compiles a series of test-cases aimed at validating the new compressible version of the code. Conclusions are finally drawn.

2 Implementation of compressibility in EllipSys

EllipSys is an incompressible viscous flow solver for which velocity and pressure are solved in a decoupled manner [1, 2, 3]. This decoupling is implemented using specific predictor-corrector algorithms such as PISO or SIMPLE (and variations from these) where the prediction step computes the velocity field from the momentum equation, and the correction step computes a pressure correction such that continuity is enforced and the predicted velocity field is accordingly corrected. In order to account for compressible effects, the approach proposed by Demirdžić *et al* [4] is implemented here.

Note also that the EllipSys code includes the option to solve the energy conservation equation which is recasted so that the static temperature is the unknown variable to be solved.

The main modifications to the original incompressible code are detailed in the following sections.

2.1 Added convective term in pressure equation

The mass flux correction enforcing the continuity equation is now added a term accounting for the varying density resulting from the pressure correction. Indeed, for a compressible fluid, any change of pressure has a direct influence on the density through the equation of state (see Eq. (3) below). The mass flux correction at an arbitrary mesh cell face reads:

$$m' = (\rho^* + \rho')(u^* + u') \approx \rho^* u^* + \rho^* u' + \rho' u^*$$

where ρ and u refer to the density and one of the velocity components, respectively, and star-quantities are predicted values while prime-quantities are correction values.

Note that the term $\rho' u'$ has been neglected in the above equation. However, in the PISO/SIMPLE algorithm and for time-true calculations, the continuity equation has to be satisfied at each time step, therefore sub-iterations are used. For steady-state calculations, the continuity equation has also to be satisfied at convergence. Thus, in both cases the pressure correction tends to zero and the above approximation has no influence on the converged results. For the same reason, the scheme used to discretize the additional term $\rho' u^*$ (i.e. the one not present for the incompressible case) has no influence on the numerical solutions when those are converged since the pressure correction should be vanishing. For stability purposes, this latter term is discretized using a standard first order upwind scheme.

The velocity correction is expressed as a function of the pressure correction according to the standard PISO/SIMPLE algorithm. The density correction is (linearly) expressed as a function of the pressure correction p' assuming an isentropic flow, yielding:

$$\rho' = \Big(\frac{\partial \rho}{\partial p}\Big) p' = \frac{p'}{c^2} \tag{1}$$

where c is the speed of sound defined as:

$$c = \sqrt{\gamma R T} \tag{2}$$

where γ and R are gas constants and T is the temperature. In the present implementation air is considered, therefore $\gamma = 1.4$ (see definition below) and $R = 287.6$ [J/(kg.K)] is the specific gas constant (for dry air).

The above equations are used to build an equation for the pressure correction in a similar way to the incompressible version, but with these new convective terms added to the mass fluxes and originating from the density correction. Note that the resulting pressure operator is no longer symmetric because of the additional terms. Therefore, the multigrid solver in the original incompressible EllipSys code, which implementation is based on this symmetry assumption, can no longer be used (at least as it is currently implemented). If one want to use the present solver in its present form, the additional terms have to be solved explicitly as part of the iterative procedure which is not recommended for numerical stability, or the operator can be symmetrized and the anti-symmetric part can be solved explicitly. Therefore, the multigrid solver is replaced by calling the existing Gauss-Seidel solver already implemented in EllipSys code. This yields to a rather slow convergence of the numerical simulations, in particular at low Mach numbers. In order to improve the numerical efficiency, the SIP

method by Stone [5] for solving linear systems is also now implemented and it proves to accelerate convergence significantly.

Finally, note that the compressible Navier-Stokes equations also require a state equation for the fluid. The classical equation for an idal gas is used:

$$p = \rho RT \tag{3}$$

2.2 New boundary conditions for velocity and pressure

Firstly, since a new variable (the density) is added to the system of equations, it is natural that additional boundary conditions must be inforced. Secondly, according to the Mach number of the flow, the flow behaves differently and may require different types of boundary conditions according to the flow regime, i.e. for subsonic or supersonic flow conditions according to the value of the inflow Mach number M define as:

$$M = U/c$$

where U in the inlet flow velocity and c is the speed of sound.

For incompressible flows, there is no boundary conditions attached to the pressure itself. Numerically, some sort of boundary conditions must be enforced when solving the pressure correction equation and these are partly determined by the chosen numerical scheme. In practice, in EllypSys homogeneous Neumann boundary conditions for the pressure correction are enforced at all boundaries and the actual pressure is extrapolated. However, in the compressible case the pressure equation is an expression of the continuity equation for the density and proper boundary conditions can be defined.

Subsonic flows

In the case of subsonic (and transonic) flows, the outlet static pressure is fixed:

$$p = pstatic\_outlet$$

using Dirichlet boundary conditions. This is a standard choice for compressible codes and the physical rationale behind this choice can be linked to the case of an open return wind tunnel which exhausts into the surrounding environment, i.e. outside the wind tunnel into ambient air.

At the inlet, it is also standard for compressible codes to enforce the inflow angle and the total pressure as specified in Eq. (4) in Section 3. In practice, during the prediction step the inflow velocity norm U is deduced from the Mach number by solving Eq. (4). This is done using the local speed of sound, or temperature using Eq (2), and the local static pressure which are both obtained from the previous (sub-)iteration.

The velocity components can then be calculated assuming that the angle at which the flow enters the computational domain is given and they are numerically enforced using Dirichlet boundary conditions. Then, in the correction step the pressure correction is enforced using homogeneous Neumann conditions in the same manner as done for the incompressible case. This configuration is obtained by using *compressible_bc*=1 in the inputs (see Section 3).

Another option for the inlet boundary conditions (*compressible_bc*=2) would be to enforce the inflow velocity (both norm and direction) using Dirichlet condition and let the total pressure evolve freely at the inlet as a result of the pressure correction equation for which boundary conditions also remain unchanged (i.e. homogeneous Neumann). It seems to be numerically unstable according to our previous experiences with the modified code, although this should be investigated further.

Supersonic flows

In the case of a supersonic flow (*compressible_bc*=3), the configuration is quite different. All quantities at the inlet are enforced using Dirichlet boundary conditions, which seems natural as no information can travel upstream of the flow. Nevertheless, Eq (4) is still used to enforce the desired total pressure. Another option would be to enforce directly the static pressure to a value chosen by the user, but it has not been implemented yet.

At the outlet, all quantities are computed using homogeneous Neumann boundary conditions assuming that an equilibrium state has been reached by the flow. Again, it seems as a natural choice as no information is travelling upstream, but it is only justified if the entire outlet boundary is immersed in fluid travelling at supersonic speed.

2.3 Other small additions to the original code

Some relatively minor or straightforward modifications to the original incompressible code due to compressibility are listed below:

- Because the fluid flow is no longer divergence-free, some additional terms are added to the shear-stress tensor. The viscous part of the momentum equation is modified accordingly. This has also an effect on the thermal production due to friction in the temperature equation (see [4] or any text book on compressible flows discussing the Navier-Stokes equations).

- If a constant specific total enthalpy is selected as input, the temperature is no longer computed using the existing EllipSys temperature module. Instead, the temperature throughout the computational domain is computed analytically by solving Eq. (6) as specified in Section 3 and using the computed velocity field.

- When solving the pressure correction, in addition to the corresponding velocity correction (as defined by the SIMPLE/PISO scheme and Rhie and Chow interpolation), it is necessary to correct the denstity. It is found that the approach to update the density that yields a numerically stable scheme is to compute the corrected density at each pressure correction using Eqs. (1) and (3) as:

$$\rho = \frac{p^* + p'/\gamma}{RT}$$

3 New input parameters specific to compressibility

Typically, numerical or physical input pameters are passed to the EllipSys code through a file, usually named 'input.dat'. The new input parameters (both keywords and their possible values) are provided in Table 1 together with their specific functions.

If p is the ambient static pressure, M the Mach number with $M = U/c$ and U being the norm of the inflow velocity, then three of the above-mentioned input parameters can be computed using the following relationships:

$$ptotal\_inlet = p\left(1 + M\frac{\gamma-1}{2}\right)^{\frac{\gamma}{\gamma-1}} \tag{4}$$

$$total\_temp\_inlet = T\left(1 + M\frac{\gamma-1}{2}\right) \tag{5}$$

$$total\_enth\_inlet = c_p T + \frac{U^2}{2} \tag{6}$$

where γ is the ratio of specific heats. For air, the pressure specific heat capacity is $c_p = 1004.703$ [J/(kg.K)] and $\gamma = c_p/c_v = 1.4$.

Note also the impact of the input parameter with keyword *total_enth_cst* (see Table 1). If specified, the energy conservation equation is not solved anymore. Instead, assuming a constant total enthalpy throughout the flow field, the temperature is calculated at each mesh cell using Eq. (6) where *total_enth_inlet* is replaced by the value assigned through the keyword *total_enth_cst*, that is:

$$T = \left(total\_enth\_cst - \frac{U^2}{2}\right)/c_p$$

Keywords	*Possible values* (Typical value)	Action
compressible	*'true'* or *'false'*	Using compressible version if 'true'
compress_reslim	*Real value* ($> 10^{-10}$)	Minimum residual for modified pressure equation to exit
compressible_bc	*Integer value* (see Section 2.2)	Type of boundary conditions: (1) Subsonic (2) Subsonic (3) Supersonic
pstatic_outlet	*Real value*	Outlet static pressure [Pa]
ptotal_inlet	*Real value* (see Eq.(4))	Inlet total pressure [Pa]
temp_inlet	*Real value*	Inlet temperature [K]
temp_farfield	*Real value*	Farfield temperature [K]
total_temp_inlet	*Real value* (see Eq.(5))	Inlet total temperature [K]
total_enth_inlet	*Real value* (see Eq.(6))	Inlet specific total enthalpy [J/kg]
total_enth_cst	*Real value* (see Eq.(6))	If specified, the temperature solver module is not used. This given specific total enthalpy is enforced throughout the entire domain instead [J/kg]

Table 1: New input parameters for compressible version of EllipSys

4 Validation of the compressible code

Several test-cases are considered hereafter in order to validate the code in several configurations.

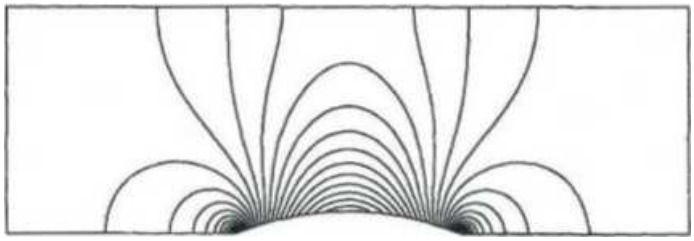
4.1 Flow in 2D channel with circular arc bump

In this section, the present code is compared with the numerical results obtained by Demirdžić *et al* [4]. They derived a compressible flow solver from an existing SIMPLE solver for incompressible flow. Note that the the present code formulation is directly inspired from their work. The test case is the inviscid flow in a 2D channel with a circular arc bump on one of its sides. Three cases are considered: (1) a subsonic case at $M = 0.5$, (2) a transonic case at $M = 0.675$, and (3) a supersonic case at $M = 1.65$. Note that for the two first cases, the height of the bump is equal to 10% of the channel width, and 4% for the third case. In all cases, the length of the bump is equal to the width of the channel. Note also that for the two first cases, the convective terms in the momentum equations are discretized using the second-order upwind scheme ('*suds*'), while a min-mod limiter is applied to this scheme in order to minimize wiggles and avoid numerical instabilities for the third case.

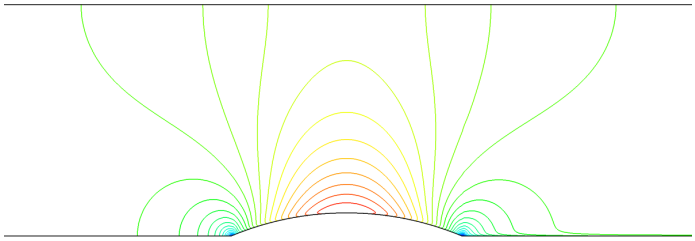
The results for the 3 test cases are displayed in the form of iso-contours of the Mach number in Figs. 1, 2 and 3, respectively. It can be noticed that the present calculations slightly differ from the results obtained by Demirdžić *et al* [4] near the bottom wall on the bump surface and in its wake. The origin of these discrepancies is explained below. However, compressible flow calculations for the same test cases using different compressible codes [6, 7] also exhibit similar patterns than those produced by the present code. Nonetheless, there is a reasonable agreement between the reference solutions from Demirdžić *et al* [4] and the present results.

The above differences are displayed in more detail by plotting the Mach number profiles along the lower and upper walls of the channel for all test-cases in Fig. 4. It can be seen that the Mach number on the lower wall as computed by the EllipSys solver does not recover its inlet value after passing the bump, whereas it does for the numerical results of Demirdžić *et al* [4]. The cause of these result discrepancies can be found in the respective mesh refinements. A computation using a coarser mesh similar to the one used by Demirdžić *et al* [4] is conducted with the EllypSys code for the subsonic case $M = 0.5$ and results are displayed in Fig. 5. Two convective schemes are also investigated: a second order upwind scheme (SUDS) and a central difference scheme (CDS). It can be seen that both the mesh refinement and the convective scheme have an influence on the wall Mach number after the bump (and also upwind of the bump for the central scheme). In any case, using the coarse mesh yields the Mach number to recover

its inlet value (or close to it) after the bump in accordance with Demirdžić *et al* [4]. The scheme also plays a role but it also influences the Mach number upwind of the bump.



(a) Demirdžić *et al* [4]



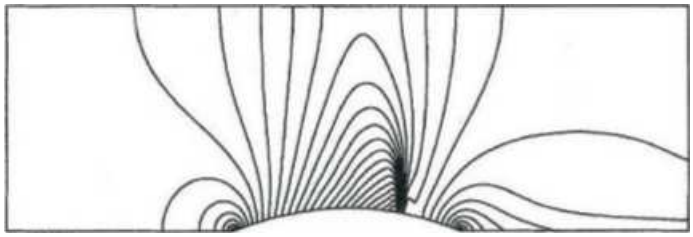
(b) EllipSys2D

Figure 1: Mach iso-contours of channel flow with 10% bump - $M = 0.5$

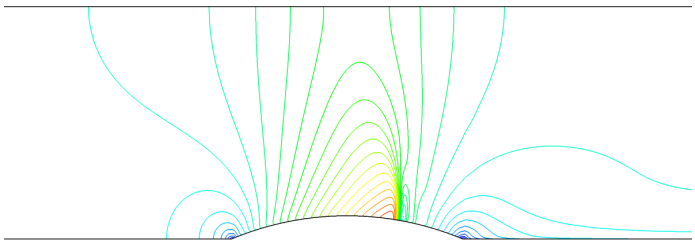
4.2 2D transonic flow around the RAE-2822 airfoil

This test-case is concerned with the 2D transonic flow around a RAE-2822 airfoil. The computational details for the reference calculation are available online [8]. The experimental data stem from a AGARD report [9]. The Mach number for this test case is equal to $M = 0.729$, the Reynolds number $Re = 6.5 \times 10^6$ and the angle of attack $\alpha = 2.31^o$. In the present calculations, the turbulence model is the SST-k-ω model and the computations are performed with (Drela's e^N model with $N_{crit} = 9$) or without transition model. The second-order upwind scheme with the min-mod limiter is used for discretizing the convective terms.

Pressure coefficient distribution, iso-contours of Mach number and pressure field calculated with the modified code are compared in Figs. 6, 7 and 8, respectively, with results from wind tunnel measurements for the former case [9] and computational results from the compressible code NPARC [8] developed at NASA for the two latter



(a) Demirdžić *et al* [4]

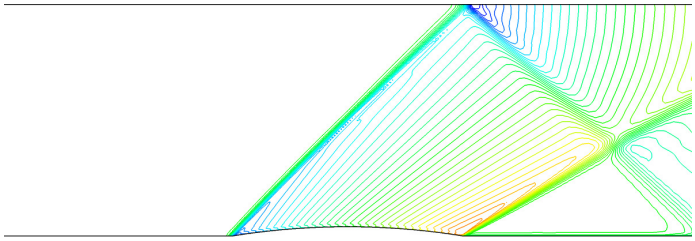


(b) EllipSys2D

Figure 2: Mach iso-contours of channel flow with 10% bump - $M = 0.675$

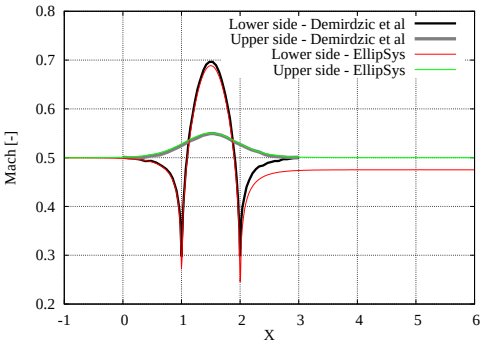
cases. The main conclusions from these comparisons are the fact that the fully turbulent calculation appears to more closely match the experimental data, and that the shock is more diffuse compared to the NPARC code results, most probably due to the scheme used in the present calculations (second-order upwind scheme with min-mod limiter) yielding to some numerical dissipation. Nevertheless, there is a general overall good agreement between the present calculations and these reference data.

(a) Demirdžić *et al* [4]

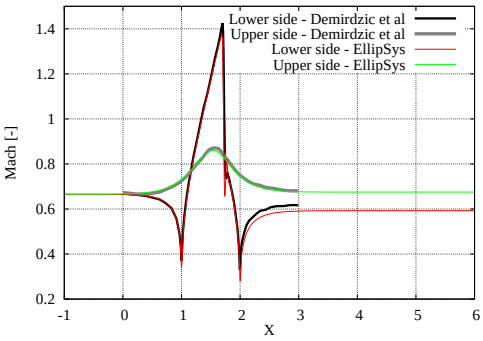


(b) EllipSys2D

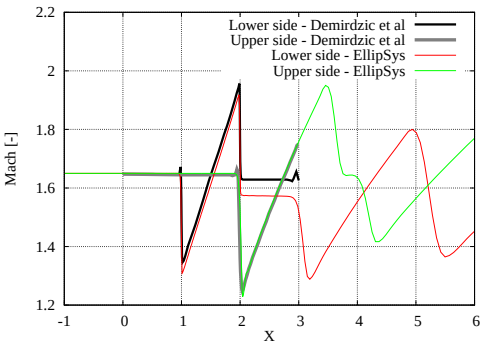
Figure 3: Mach iso-contours of channel flow with 4% bump - $M = 1.65$



(a) Bump = 10% - $M = 0.5$



(b) Bump = 10% - $M = 0.675$



(c) Bump = 4% - $M = 1.65$

Figure 4: Mach profiles along lower and upper walls of channel

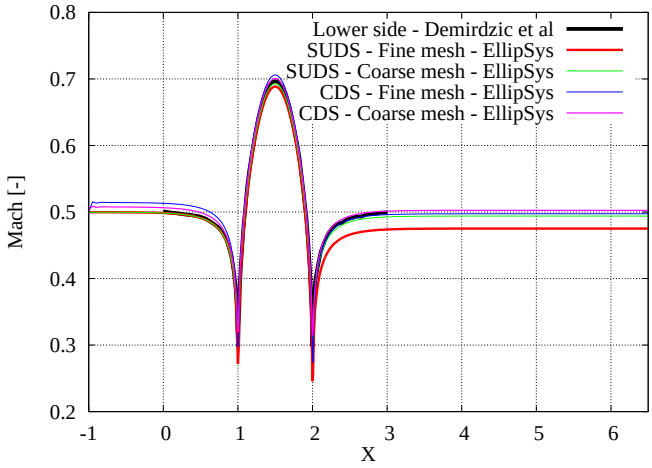


Figure 5: Mach profiles along lower wall of channel for various mesh refinements and convective schemes (Bump = 10% - $M = 0.5$)

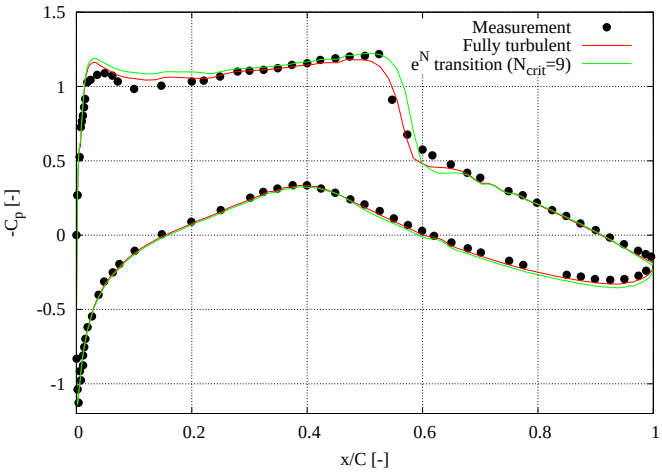
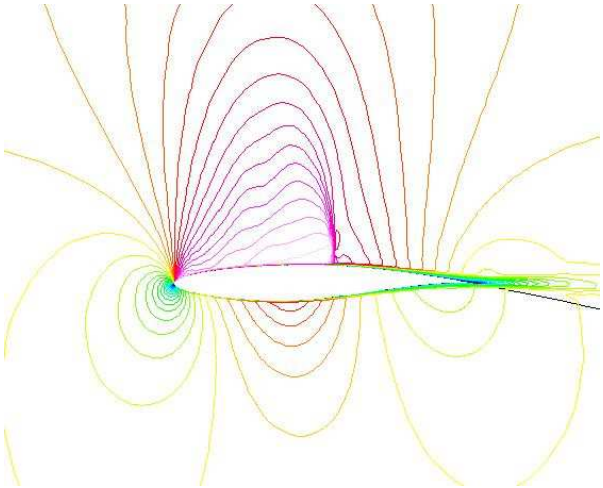
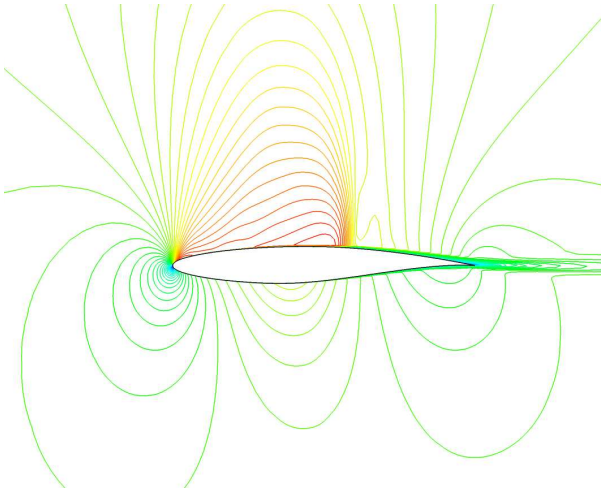


Figure 6: Pressure coefficients for RAE-2822 airfoil at $M = 0.729$, $Re = 6.5$M and $\alpha = 2.31^o$

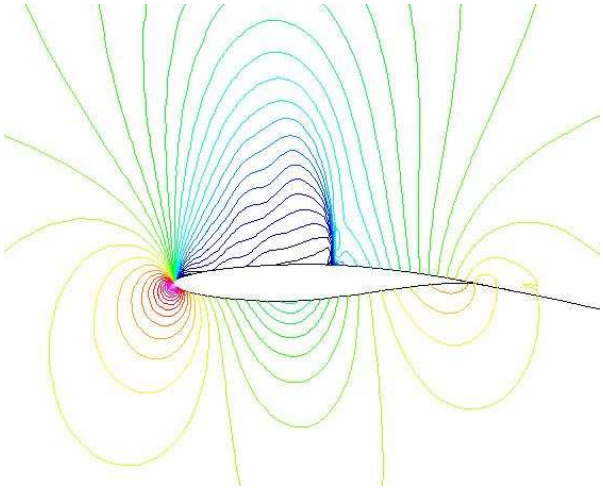


(a) NPARC [8]

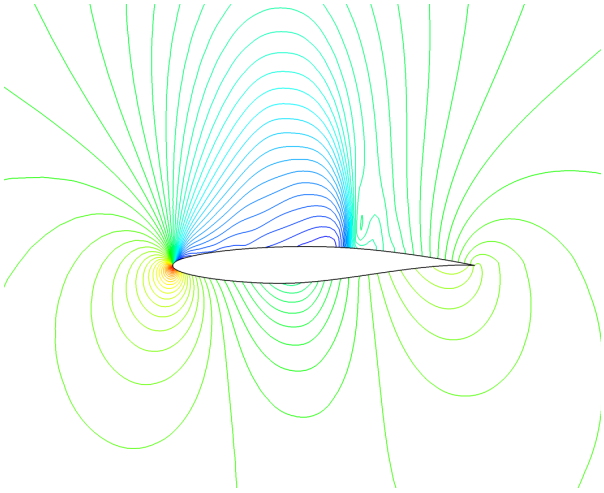


(b) EllipSys2D

Figure 7: Mach iso-contours of RAE-2822 airfoil flow at $M = 0.729$, $Re = 6.5$M and $\alpha = 2.31^o$



(a) NPARC [8]



(b) EllipSys2D

Figure 8: Pressure iso-contours of RAE-2822 airfoil flow at $M = 0.729$, $Re = 6.5$M and $\alpha = 2.31^o$

5 Conclusions and perspectives

In this report, modifications of the EllipSys2D code implemented in order to simulate compressible flows are reported. The main change as far as the user is concerned is the possibility to activate the compressible part of the code using a series of new input parameters. The latter are detailed in this report and it should be relatively straightforward to use this new functionality.

The code is verified and numerical results are compared with existing reference solutions. These comparisons show that the results are qualitatively correct and that the code is behaving soundly, at least in the range of conditions for which these tests are performed, which are well beyond wind turbine applications (such as hypersonic flows).

The next step will be to include these code modifications into the 3D version of EllipSys. It is believed that it should be rather easy since these modifications only involve a small part of the overall 2D code. Nevertheless, it can be expected that computing time and efficiency might become a limiting issue when solving 3D flow configurations, and that it may be necessary to implement a multigrid solver for the pressure equation that can handle a non-symmetric operator.

Acknowledgments

This work was supported by the AVATAR project (as part of the European Energy Research Alliance, EERA, grant agreement No. FP7-ENERGY-2013-1/no. 608396) carried out under the FP7 program of the European Union, as well as the 'Rain Erosion Tester for Accelerated Test of Wind Turbines' project funded by EUDP (Energy Technology Development and Demonstration Program, Jr. nr. 64015-0045) of Energistyrelsen (the Danish Energy Agency).

References

[1] Michelsen J. A., *Basis3D - A Platform for Development of Multiblock PDE Solvers finite volume method for predicting flows at all speeds*, Tech. Rep. AFM 92-05, Technical University of Denmark, Lyngby, Denmark (1992)

[2] Michelsen J. A., *Block Structured Multigrid Solution of 2D and 3D Elliptic PDE's*, Tech. Rep. AFM 94-06, Technical University of Denmark, Lyngby, Denmark (1994)

[3] Sørensen N. N., *General Purpose Flow Solver Applied to Flow over Hills*, PhD Thesis, Tech. Rep. Risø-R-827(EN), Risø National Laboratory, Roskilde, Denmark (1995) 92-05, Technical University of Denmark, Lyngby, Denmark (1992)

[4] Demirdžić, I., Lilek, Ž and Perić, M., *A colocated finite volume method for predicting flows at all speeds*, Int. J. Numer. Methods Fluids, **16**, pp. 1029–1050 (1993)

[5] Stone, H. L., *Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations*, J. of Numer. Anal., **5**(3), pp. 530-538 (1968)

[6] Zhou, J., Zhang, Y. and Chen, J. K., *Numerical simulation of compressible gas flow and heat transfer in a microchannel surrounded by solid media*, Int. J. of Heat and Fluid Flow, **28**, pp. 1484–1491 (2007)

[7] Mahadi, A., Kermani, M. J. and Khazaei, H., *Improvement of a solution of inviscid compressible flows using a mixed wall boundary condition*, Engineering Applications of Computational Fluid Mechanics, **9**(1), pp. 126–138 (2015)

[8] RAE2822 Transonic Airfoil (2008), Retrieved from: https://www.grc.nasa.gov/www/wind/valid/raetaf/raetaf.html

[9] Cook, P. H. and McDonald, M. A. and Firmin, M. C. P., *Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements*, Experimental data base for computer program assessment, AGARD Advisory Report, AGARD-AR-138 (1979)

DTU Wind Energy is a department of the Technical University of Denmark with a unique integration of research, education, innovation and public/private sector consulting in the field of wind energy. Our activities develop new opportunities and technology for the global and Danish exploitation of wind energy. Research focuses on key technical-scientific fields, which are central for the development, innovation and use of wind energy and provides the basis for advanced education at the education.

We have more than 230 staff members of which approximately 60 are PhD students. Research is conducted within 9 research programmes organized into three main topics: Wind energy systems, Wind turbine technology and Basics for wind energy.

DTU Wind Energy
Department of Wind Energy
Technical University of Denmark

Main address:
Campus Risø
Building 118
Frederiksborgvej 399
DK-4000 Roskilde
Tlf. (+45) 46 77 50 85

www.vindenergi.dtu.dk
email: info@vindenergy.dtu.dk